Refine Search

Search Results -

Terms	Documents
L8 and L7	3

[illegible] 12-14-04

Database:
US Pre-Grant Publication Full-Text Database
US Patents Full-Text Database
US OCR Full-Text Database
EPO Abstracts Database
JPO Abstracts Database
Derwent World Patents Index
IBM Technical Disclosure Bulletins

Search: L11

Refine Search

Recall Text   Clear   Interrupt

Search History

DATE: Tuesday, December 14, 2004   Printable Copy   Create Case

Set Name side by side	Query	Hit Count	Set Name result set
DB=PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=OR			
L11	L8 and L7	3	L11
L10	L9 and L7	3	L10
L9	717/$.ccls.	7547	L9
L8	707/$.ccls.	23948	L8
L7	upgrad$3 same ((intermediate or secondary) near version$)	13	L7
DB=USPT; PLUR=YES; OP=OR			
L6	L5	22	L6
DB=PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=OR			
L5	L1 and upgrad$	64	L5
L4	L2 and upgrad$	986	L4
L3	updat$3 near message	5939	L3
L2	(updat$3 or chang$3) near message	11009	L2
L1	(intermediate or secondary) near version$	667	L1

END OF SEARCH HISTORY

L10: Entry 2 of 3 File: USPT Mar 19, 2002

DOCUMENT-IDENTIFIER: US 6360363 B1
TITLE: Live upgrade process for object-oriented programs

Brief Summary Text (18):
During phase (2), however, which is the offline preparation phase, a mechanism of the invention prepares an intermediate program P' that contains an intermediate version of each of the program modules to be upgraded. The intermediate version of a program module contains both the old version used in P and the new version used in P". The intermediate version of the program module is used to ensure that the program can continue to operate without disruption of service while it is being upgraded. The mechanism inserts program code into the intermediate version that allows the module to switch from the old version to the new version, converting the state of the old version into the state of the new version as it does so.

Drawing Description Text (3):
FIG. 1 is a flow diagram showing the upgrading of an object, from the old version C.sub.1, at the top to the new version C.sub.1 " at the bottom, with the intermediate version C.sub.1 ' in the center.

Detailed Description Text (26):
During phase (2), the offline preparation phase, the "Preparer" mechanisms of the invention prepare a program P' that contains intermediate versions of the classes to be upgraded. These intermediate versions are used to ensure that the computer program can continue to operate without disruption while it is being upgraded.

Detailed Description Text (28):
The detailed description of the live upgrade process given below considers first, in Section 1, the mechanisms that the Upgrader uses in phase (3) to upgrade a single object of a class with no changes to the signatures of the methods of the class. Then, in Section 4, it addresses the mechanisms that the Upgrader uses in phase (3) to upgrade the objects of several classes, each with several object instances and with changes to the signatures of the methods of the classes as described in Section 4. Finally, in Section 6, the description considers the mechanisms that the Preparer uses in phase (2) to prepare the intermediate versions of the classes with the assistance of the programmer.

Detailed Description Text (54):
If, instead, the Upgrader upgrades the objects C.sub.1, C.sub.2 and C.sub.3 at different points in time, upgrading each object when it is individually inactive, it is possible that C, has been upgraded and is invoking the methods of D.sub.1 and E.sub.1 using the new method signatures, while C.sub.2 has not yet been upgraded and is still invoking the methods of D.sub.1 and E.sub.1 using the old method signatures. The Preparer mechanisms of the invention construct intermediate versions of the objects of the classes D and E that support both the old method signatures and the new method signatures concurrently. The means by which those intermediate versions are constructed, by interaction with the programmer during the preparation for the upgrade, is described in Section 6.

Detailed Description Text (58):
Until all objects have been upgraded, as indicated by a test for inactive objects

at step 109, at step 105 the algorithm selects for upgrading an object of a class in the coordinated upgrade set, provided that (1) all objects of classes (step 107) in the coordinated upgrade set that it invokes have been upgraded to support both the old and the new methods (step 108), and (2) it is inactive (step 109). If the object is an instance of a top class (step 110), the Upgrader upgrades the object to its final version at step 112. Otherwise, the Upgrader upgrades the object to an intermediate version at step 111. In both cases, this replacement is performed by the mechanisms described in Section 2 and Section 3, and shown in FIG. 2A and FIG. 2B, for the upgrading of a single object.

Detailed Description Text (60):
As shown in FIG. 7, the first step involves replacing the object D.sub.1 by an object D.sub.1 ' that supports both the old methods and the new methods (step 121). Next, the object E.sub.1 is replaced similarly by E.sub.1 ' at step 122. At this stage, the methods of both D.sub.1 and E.sub.1 can be invoked using either the old or the new signatures and, thus, the objects C.sub.1, C.sub.2 and C.sub.3 can be replaced by C.sub.1 ", C.sub.2 " and C.sub.3 ", respectively, in any order at steps 123,124 and 125. Because C is a top class, the signatures by which the objects of class B invoke methods of the objects of class C are not changed; thus, there is no need to construct intermediate versions of the objects of class C. Once all of the objects of class C have been replaced at step 125, D.sub.1 ' is replaced by D.sub.1 " at step 126 and then E.sub.1 ' is replaced by E.sub.1 " at step 127 to complete the upgrade. Each of the replacements of C.sub.1, C.sub.2 and C.sub.3 and the replacements of D.sub.1 and E.sub.1 are performed using the mechanisms described in Section 2 and Section 3, and shown in FIG. 2A and FIG. 2B, for the upgrading of a single object.

Detailed Description Text (63):
At this point, the Preparer identifies the classes that have been changed and must be upgraded at step 133, and also determines the coordinated upgrade sets and their top classes at step 134. For each of the coordinated upgrade sets (step 135) and for each class in such a set (step 136), the Preparer determines whether the class is a top class (step 137). If the class is a top class, the Preparer generates an intermediate class (step 138). Otherwise, the Preparer generates a primary intermediate version of a class (step 139) and also the first intermediate version (step 118) and the last intermediate version (step 119). The objects of the classes in each of these coordinated upgrade sets are upgraded in the order determined by the algorithm for the Upgrader, described in Section 5 and shown in FIG. 6A and FIG. 6B.

Detailed Description Text (72):
For a non-top class, three intermediate versions of a class must be generated. The middle one is referred to as the primary intermediate version (step 139), and the other two are referred to as the first intermediate version (step 118) and the last intermediate version (step 119). All objects of the primary intermediate version must be capable of executing methods that were invoked by objects of both the old and the new versions of the classes, as shown in FIG. 7. To permit the upgrade from the old version to the primary intermediate version, and the upgrade from the primary intermediate version to the new version, two additional intermediate versions (the first intermediate version and the last intermediate version) are used to facilitate the atomic switchover shown in FIG. 3. As indicated above, the first intermediate version and the last intermediate version are generated by the algorithm described in Section 7 and shown in FIG. 9. The primary intermediate version (step 139 in FIG. 8) is generated by the algorithm with the assistance of the computer programmer, as described in Section 8 and shown in FIG. 10.

Detailed Description Text (75):
The attributes of the primary intermediate version of a class, defined by the programmer at step 163, are derived from those of the old and new versions of the class. Typically, but not always, the attributes are those of the new version of

the class (step 164). If so (step 165), the methods of the primary intermediate version, that are invoked by objects of new versions of other classes, can be derived directly from the methods of the new version of the class being upgraded. The modified methods, defined by the programmer, can be invoked by objects of the old versions of other classes (step 166). If the attributes of the primary intermediate version are not those of the new version of the class, the programmer must define both the methods to be invoked by objects of the new versions of other classes at step 167, and also the methods to be invoked by objects of the old versions of other classes at step 168. The Preparer checks the correspondence between the signatures of the methods of the primary intermediate version of the class prepared by the programmer and the signatures of the corresponding methods of the old and new versions of the class at step 169.

Current US Original Classification (1):
717/170

CLAIMS:

3. A method as recited in claim 2, wherein an object of a class can be upgraded only when all objects of classes in the coordinated upgrade set, that are invoked by the object, have been upgraded to their primary intermediate versions.

5. A method as recited in claim 2, wherein a class in a coordinated upgrade set can be upgraded only when all classes in the coordinated upgrade set, whose objects are invoked by objects of the class, have been upgraded to their primary intermediate versions.

Previous Doc Next Doc Go to Doc#